Opened: Monday, 15 January 2024, 12:00 AM
Due: Friday, 26 January 2024, 11:59 PM

Image Basics: know how to access pixels from an image matrix, compute a gray scale histogram and learn to equalize, compare and match histograms. This notebook will be of some help here.

**This assignment is worth 9% of overall weight.**

Submission:
You are provided with a single source file called A1_submission.py along with 3 images to be used as input in your code. You need to complete the four functions indicated there, one for each part. You can add any other functions or other code you want to use but they must all be in this same file.

You need to submit **only** the completed A1_submission.py.

Part I (20%): Complete function **part1_histogram_compute()** to accomplish the following:

1. Read the grayscale image called *test.jpg*.
2. Write **your own code** to compute a 64-bin gray scale histogram of the image. You **cannot** use built in histogram functions from any library (e.g. *numpy.histogram, scipy.stats.histogram, skimage.exposure.histogram*, *opencv.calcHist*, etc) for this.
3. Plot the histogram.
4. Also, call Numpy histogram function to compute 64-bin histogram for the same image. Plot it side by side with yours to show that they are identical.

Expected output:
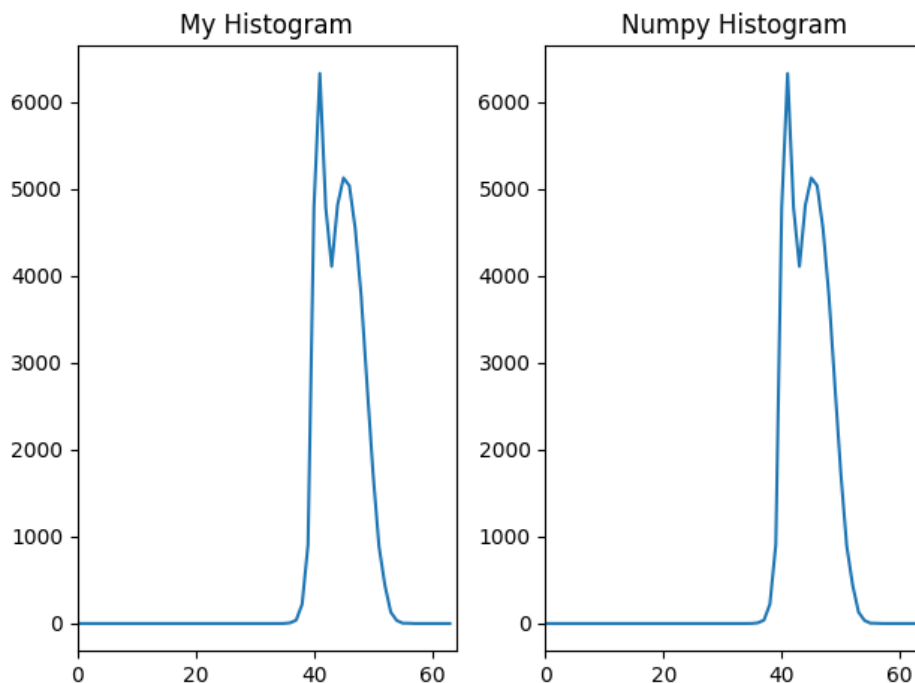
What is a 64-bin histogram?: 8-bit images have pixel values that range from 0 to 255. This means that each bin will have 4 values, i.e. the first bin will have the count of pixel values 0-3, the second bin will have that for 4-7, 3rd: 8-11, ..., 64th bin: 252-255.

Part II (30%): Histogram equalization:

?

Complete function **part2_histogram_equalization()** to perform a 64-bin grayscale histogram equalization on the same *test.jpg* image used in the last part. You need to plot the original image, its 64-bin histogram, the image after histogram equalization, and its 64-bin histogram.

You are **not** allowed to use the Skimage functions, i.e., *exposure.histogram*, *exposure.equalize_hist,* or any equivalent functions from any other library for this part.
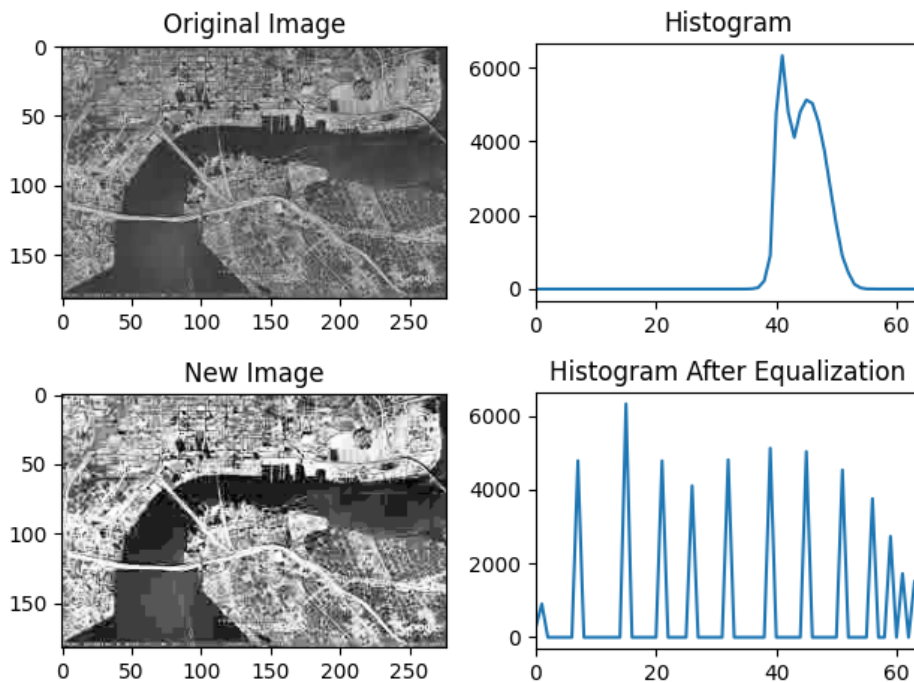
Tutorials on histogram equalization:

https://www.youtube.com/watch?v=GWCB3pKi2ko

https://www.tutorialspoint.com/dip/histogram_equalization.htm

https://en.wikipedia.org/wiki/Histogram_equalization

Expected output:

Part III (10%): Histogram comparing:

Complete function **part3_histogram_comparing()** to compare the 256-bin histograms of two images *day.jpg* and *night.jpg*. You will need to read both images, convert them to grayscale, compute their 256-bin histograms and print the Bhattacharyya Coefficient of the two histograms.

You can use numpy histogram function to compute these histograms.

Expected output:

Bhattacharyya Coefficient: 0.8671201323799057

Part IV (40%): Histogram specification (also called histogram matching):

Complete function **part4_histogram_matching()** to match the histograms of the same two images *day.jpg* and *night.jpg* from part 3.

You need to implement the algorithm given on the last page of the histogram equalization example on eclass:

Inputs: (1) Input image: I, (2) gray scale range:{0,,,K-1}, (3) reference histogram
Output: (1) Output image J

Step 1: Compute PA(a), i.e., the normalized cumulative input histogram

Step 2: Compute PR(a), i.e., the normalized cumulative reference histogram

Step 3: Compute this function A[a] = invPR(PA[a])
        a' = 0
        for a in {0,1,...,K-1}
            while PA[a] > PR[a']
                a' +=1
            A[a] = a'

Step 4: for i in the range of height of I
             for j in the range of the width of I
                  a = I[i, j]
                  J[i, j] = A[a]

You can use numpy histogram function to compute the histograms.

(a) (30%) Grayscale:

Read both images, convert them to grayscale, and match the 256-bin histogram of the day image to that of
the night image to generate a new grayscale image that should be a darker version of the day image. Show the grayscale day,
night, and matched day images side by side.

(b) (10%) RGB:

Repeat the grayscale histogram matching process from part (a) with each channel of the two images and put together the
resultant matched channels into a new RGB image. You can also use the single-intensity mapping obtained from the grayscale
images in (a) to match each of the three channels as suggested in the third tutorial below. Show the RGB day, night, and matched
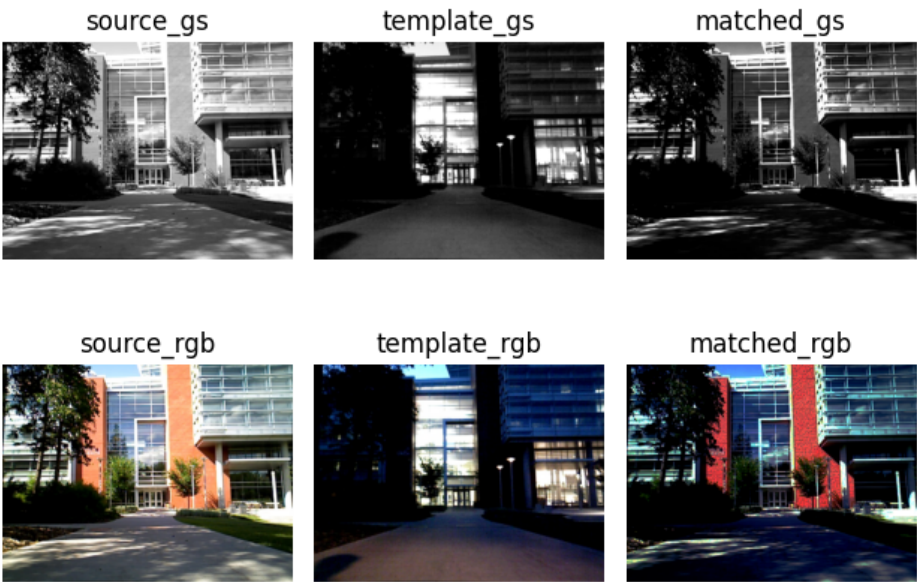day images side by side.

Tutorials on histogram matching:

https://towardsdatascience.com/histogram-matching-ee3a67b4cbc1

https://en.wikipedia.org/wiki/Histogram_matching

http://paulbourke.net/miscellaneous/equalisation/

Expected output:

Note about image paths in your submission especially if you use colab to complete your assignment:

Please make sure to remove custom image paths from **_imread_** calls and leave only the image name.

For example, absolute paths like this in your colab notebook:

skimage.io.imread("/content/drive/MyDrive/206/A1/test.jpg")

shouuld be replaced with

skimage.io.imread("test.jpg")

in _A1_submission.py_ before submitting it.

_A1_submission.py_ is expected to run as a standalone file with the assumption that all input images are present in the same folder as the file itself.

Any errors caused by the presence of absolute image paths will be penalized.

Add submission

Submission status

Attempt number	This is attempt 1 (1 attempts allowed).
Submission status	No submissions have been made yet
Grading status	Not graded
Time remaining	5 days 4 hours remaining
Last modified	-

Submission comments	▶ Comments (0)